Lec 1 Compilers

→ what is Compilers?
→ Phases of Compilers?
→ Function of Phases
→ The Programs we use to Process it.

} → our goals.

Translators → do the same function of Compilers.
→ Types of Translators? their functions?

Preprocessor :-

ـ يأخذ micro inst ويجهزها بتاعها.

in Programming #include ---- السطر ده بتاع ال (Preprocessor)

← بعض كومبايلرز يقدر يعمل (translation) من (high level langauge) لل (another high level language)

→ some Compilers Can give us executable Code (ready) or assembly Code (need some operations to be ready)

→ Translator has to get two Programs do the same function to translate between them.

Types of Translator

→ **Pre Processor**

← يحول من (level language) to (level language)

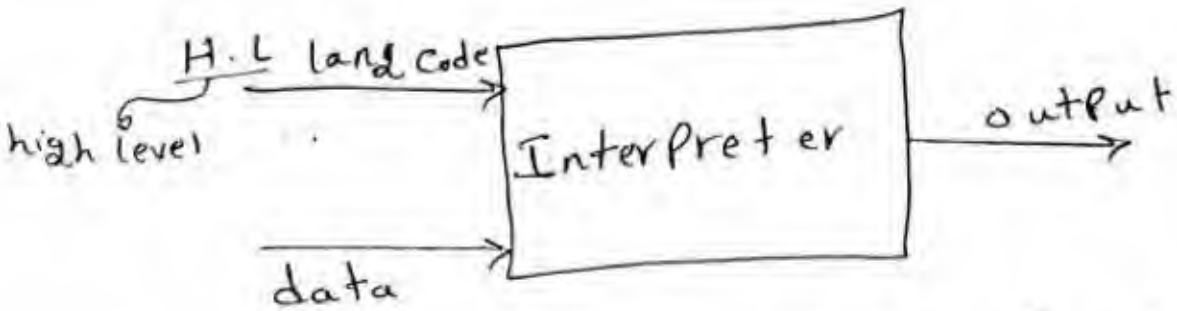
→ **Assembler**

→ It is one to one ↓

كل (assembly inst.) يقابل (one instruction)

(Disassembler) ← يقوم بعكس (assembler)

← (Assembler) مش معنى انه يطلع (o/p) إحنا نعمله (run) على (machine) ونطلع o/p.



→ difference between Compiler & interpreter

↳ on pdf slides.

(run) أسرع و أول (interpreter) ←

بس ← (in total) ال (Compiler) أسرع.

← لو ال (interpreter) قابل (error) يؤثر على جزء

يتوقف تماماً أما ال (Compilers) فيكمل العمل.

Compilers

→ classified for instruction

a) single Pass.

b) multiple Pass.

→ classified with Considration of optimization:-

a) Global b) local

→ There are anothers do the same function or similar function to the compilers.

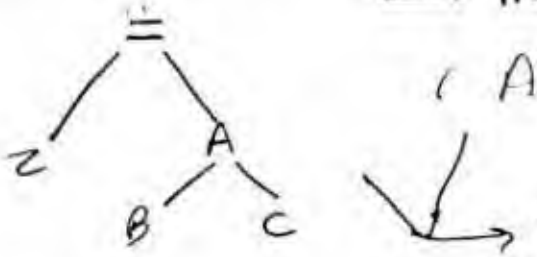
Like cross-compiler, source-to-source.

→ preprocessor is similar to source-to-source in the definition.

→ differeence between source-to-source compiler and preprocessor ⟹ report.

Architecture of compilers

→ front-end → away from machine
→ machine independent.



(ADD (A, B, C)

→ intermediate representation
↳ not related to machine.

→ Analysis do in three parts.

Analysies → lexical (scanner)
→ syntax
→ semantic

Front end → Analysis
↳ intermediate code representation.

← القياس ما بين (Front end) و (back-end)
هو علاقته بال (machine).

back-end → if we work with local optimization.

→ what is Parsing ??

→ syntax-analysis ⇒ it's 2nd name is Parsing.

→ machine independent → global optimiser.

→ machine dependent → local "

⇒ Phases we will study more:-

* lexical
* syntax
* Code Generator
* machine dependent code optimiser.

→ Compiler Function

→ transform from high level lang. to low level lang.

→ handling for error.

→ build symbol table.

* Symbol table

← أي (variable) ← يخزنه في (symbole table) يحجز مكان في الذاكرة ثم يتم تخزين الـ (symbole table) في الذاكرة أي أنه يأخذ مكان في الـ (memory).

→ search for it.

→ Lexical (scanner)

الـ tokens هنا بعني الـ (word) لكن في البرنامج فقط.

→ takes statement by statement.

تشوف الـ (Lexical analysis) (command) فيا معناه إنه مش بيشيله (Preprocessor).

→ lexemes must be meaningful as a language.

→ syntax analysis

→ check source code for errors.

→ give us parsal tree or atoms as o/p.

→ search of the sentence's structure.

intermediate lang → machine independent.

← هذه المحاضرة عبارة عن ملاحظات تساعد على فهم ال pdf فقط.

7 Lec 1